文档梳理之配件部分

12. 配件

12.1 游戏枪

12.1.1 基本描述

游戏枪是乐视跟深圳市欢创科技有限公司(TVplay)进行合作的一个项目，游戏枪目前有两种，一种是步枪配合专用的摄像头来使用，一种是手枪配合专用的接收器来使用。电视中针对两种枪做相应的适配，游戏中结合相应的 sdk 能够识别到枪瞄准的位置并画出光标，能够识别射击、移动等指令，从而达到游戏枪来操控游戏的目的。

12-1 步枪

12-2 步枪配套的摄像头

12-3 手枪

12-4 手枪接收器

12.1.2 工作原理

以步枪为例，步枪上有三个红外信号发射点，摄像头中有红外信号接收器，枪的位置及射击等动作都是通过红外信号来传递的。摄像头的驱动负责完成上述算法，游戏 sdk 通过 jni 直接跟驱动进行交互然后提供给游戏 app 以接口，最后由游戏 app 来完成 UI 层面的响应。

手枪跟步枪原理一致，只是外在形式上不一样。

12.1.3 代码位置及基本功能描述

1)vendor/mstar/kernel/drivers/idong200alg
驱动代码，负责识别游戏枪位置、射击、移动等。

2)device/mstar/mangosteen/preinstall/modules/idong
其 中 libHidAbsMouse.so libidongjni.so libxEyeEngine.so libxeyezCmdApi.so 是跟驱动配套的几个库；其中 CheckIdongStatus 负责控制摄像头的灯；其中 init.toltech.sh 负责插入驱动并做一些配置。

3）vendor/letv/network/stvrootservice/
在打开 selinux 时，有些操作会被阻止，而由于方案设计的原因改动起来会有很大工作量，因此在系统中提供了这个 root 用户启动的 service，部分操作通过此 service 来执行，从而避免了因部分操作被 selinux 阻止而无法正常工作的问

题。

4)vendor/letv/network/LetvTVplayManager

这是一个充当 service 角色的 app,TVplay 会同其他游戏公司来共同开发支持游戏枪的游戏,游戏手枪游戏的 app 没有打包游戏 sdk 的话就会用到这个 service。

5)device/mstar/mangosteen/init.maserati.rc

负责监听一个 property，待游戏枪摄像头及接收器是否插入并且此 property 被置位后再执行插入驱动、启动 stvrootservice、启动 LetvTVplayManager 等操作。

6)frameworks/base/services/usb/java/com/android/server/usb/UsbHostManager.java

负责监听摄像头和接收器插入事件，发现后设置 property 以便插入驱动、启动 service

7) device/mstar/common/libraries/gamecursor/

device/mstar/common/libraries/doublegamecursor/

frameworks/native/services/surfaceflinger/SurfaceFlinger.cpp

frameworks/native/services/surfaceflinger/DisplayHardware/HWComposer.cpp

hardware/mstar/hwcomposer2/hwcomposer.cpp

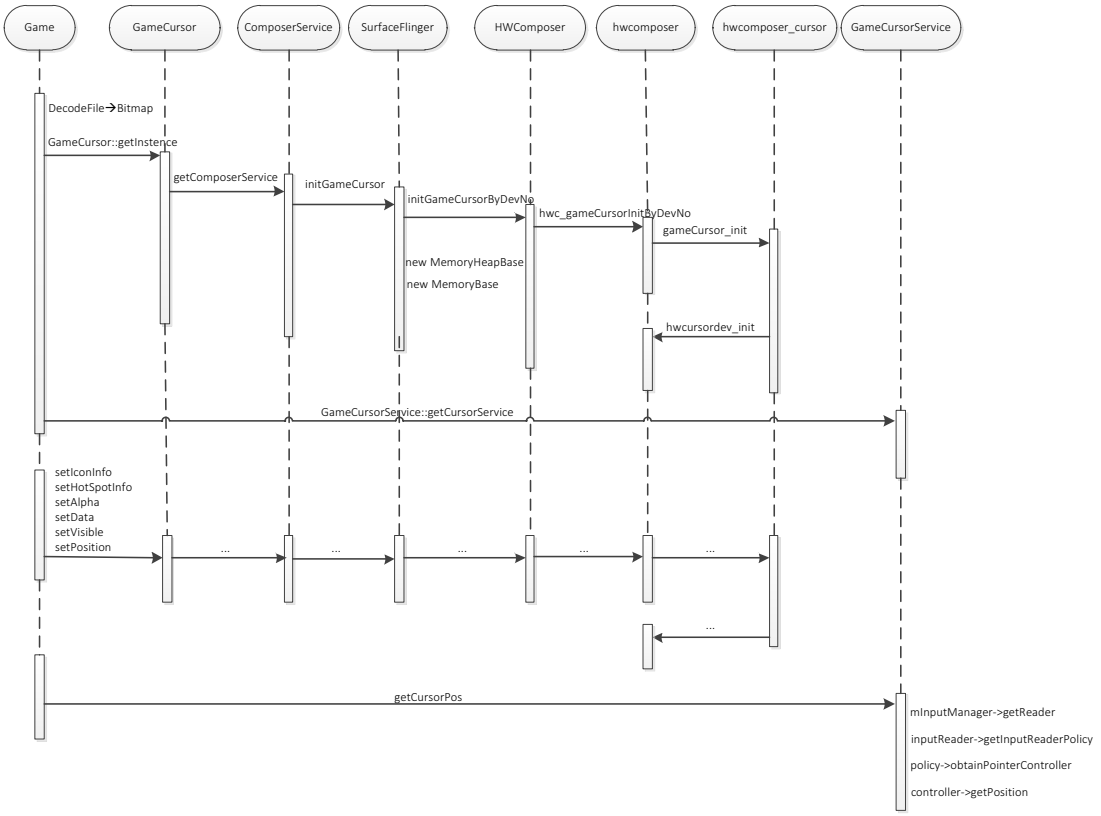
hardware/mstar/hwcomposer2/hwcomposer_cursor.cpp

一系列的接口封装，提供了 cursor 的绘制及设置大小、透明度、位置等的接口

12.1.4 软件逻辑

通过红外识别到枪的位置及射击等操作由第三方的非开源代码实现，部分开源的代码也没有仔细研究了。

有关光标的控制部分大致的调用流程如下：



12-5 光标控制流程图

12.2 手柄

12.2.1 基本描述

手柄和台杆都是 ODM 的产品，厂商提供样品供我们进行适配，会根据我们这边的软硬件测试结果进行调整和改进。BSP 这边的工作主要针对按键、扳机、摇杆的键值进行适配，另外如果支持蓝牙连接还会涉及蓝牙配对的验证工作。

12.2.2 工作原理

常见的手柄基本都支持 2.4G 方式进行连接，部分手柄支持蓝牙连接。2.4G 连接方式会提供 usb 接收器，插到电视上有供电后接收器会通过 2.4G 与手柄进行通信，接收器通过 usb 驱动将手柄的按键事件通知 kernel，kernel 再将事件上抛到 inputFlinger、Framework、app。

蓝牙连接模式下不同的地方在于在协议栈中解析数据后向 kernel 注入 HID 事件的过程。

适配过程中主要关心的是这款设备的 VendorId、ProductId、按键的 scancode、摇杆及扳机键的 axis-id、部分按键的字符映射是否正常及可能会涉及的自定义按键。

12.2.3 代码位置及基本功能描述

1) frameworks/base/data/keyboards/
device/mstar/mangosteen/preinstall/keylayout/
kl 文件和 kcm 文件大部分在第一个目录下，一部分在第二个目录。inputFlinger 将 kernel 传上来的事件中 scancode 值做一次映射，转换成 keycode 后供 framework 及 app 使用，使得不同的手柄、台杆、遥控器设备在 framework 和 app 层面透明。而这个映射规则由 kl 和 kcm 文件中将 scancode 及 keycode 的标签名一一对应写明。

2) frameworks/base/api/current.txt
frameworks/base/api/system-current.txt
frameworks/native/include/android/keycodes.h
frameworks/native/include/input/InputEventLabels.h
frameworks/base/services/core/java/com/android/server/policy/PhoneWindowManager.java
frameworks/base/core/java/android/view/KeyEvent.java
frameworks/base/data/keyboards/Vendor_xxxx_Product_xxxx.kl
新定义一个键值(keycode)涉及到的文件

12.2.4 适配流程

1）获取设备的 VendorId，ProductId。需要留意有些设备可能支持有线、2.4G、蓝牙这三种连接方式中的多种，每种连接情况下的 VendorId 和 ProductId 都是不同的。

```
I: Bus=0003 Vendor=0101 Product=1d79 Version=0110
N: Name="MY-POWER LeGPD-100"
P: Phys=usb-mstar-1.2/input1
S: Sysfs=/devices/platform/Mstar-ehci-1.0/usb2/2-1/2-1.2/2-1.2:1.1/input/input16
U: Uniq=
H: Handlers=kbd event12
B: PROP=0
B: EV=1f
B: KEY=4837fff 72ff32d bf544446 0 0 1 20c10 b17c000 267bfa d941dfed 9e1680 4400 0 10000002
B: REL=40
B: ABS=1 0
B: MSC=10

root@mstarnapoli:/ #
root@mstarnapoli:/ # cat /proc/bus/input/devices
```

12-6 VendorId、ProductId 查看方式

2）根据查看到的 VendorId、ProductId 创建 kl 文件，比如：Vendor_0101_Product_1d79.kl，也有可能两款手柄或者其他设备的 VendorId、ProductId 一样，而其中部分按键的 scancode 不一致，则可以根据 VersionId 来进一步区分，比如 Vendor_0101_Product_1d79_Version_0110.kl

```
root@mstarnapoli:/ # cat /system/usr/keylayout/Vendor_0101_Product_1d79.kl
# Copyright (C) 2011 The Android Open Source Project
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

#
# For devices:
# 1. Letv LeGPD-100 Joystick
#

key 304   BUTTON_A
key 305   BUTTON_B
key 307   BUTTON_X
key 308   BUTTON_Y
key 310   BUTTON_L1
key 311   BUTTON_R1
key 312   BUTTON_L2
key 313   BUTTON_R2
key 158   BACK
key 116   GAMECENTER
key 315   BUTTON_START
key 317   BUTTON_THUMBL
key 318   BUTTON_THUMBR
key 377   LETV
key 172   HOME
key 139   MENU

# Left and right stick.
# The reported value for flat is 128 out of a range from -32767 to 32768, which is absurd.
# This confuses applications that rely on the flat value because the joystick actually
# settles in a flat range of +/- 4096 or so.
axis 0x00 X flat 4096
axis 0x01 Y flat 4096
axis 0x02 Z flat 4096
axis 0x05 RZ flat 4096

axis 0x09 RTRIGGER
axis 0x0a LTRIGGER

# Hat.
axis 0x10 HAT_X
axis 0x11 HAT_Y
root@mstarnapoli:/ #
```

12-7 kl 文件内容

3）通过 getevent 获取具体按键的 scancode 值，及摇杆的 axis-id。

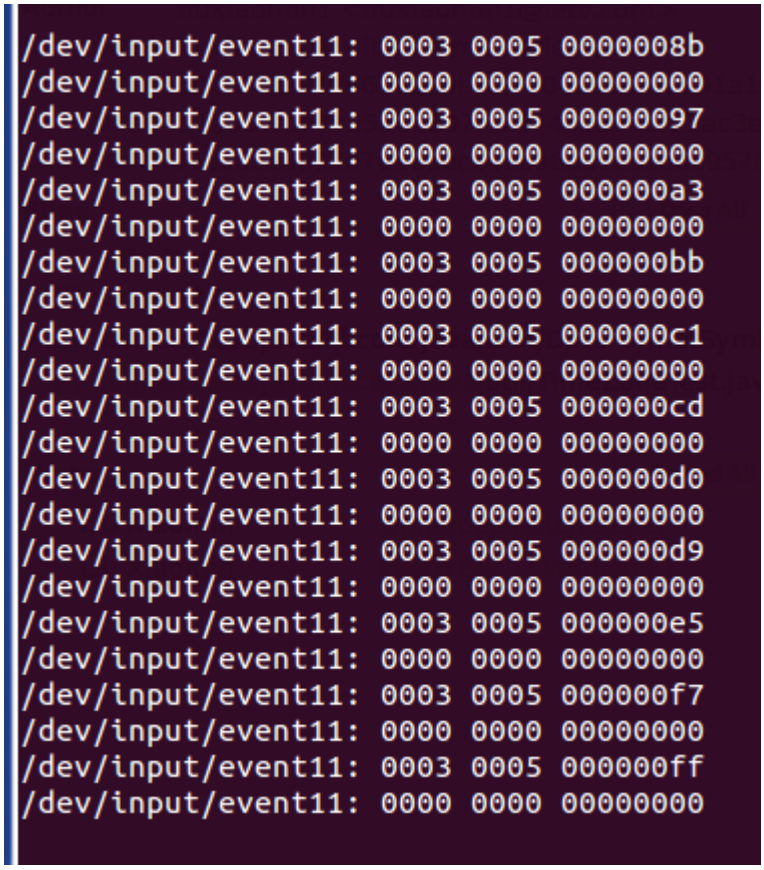
按键 A，从 getevent 中得到 scancode：0x130，对应 kl 中的 key 304 BUTTON_A，其他按键同理。

```
/dev/input/event11: 0004 0004 00090001
/dev/input/event11: 0001 0130 00000001
/dev/input/event11: 0000 0000 00000000
/dev/input/event11: 0004 0004 00090001
/dev/input/event11: 0001 0130 00000000
/dev/input/event11: 0000 0000 00000000
```

12-8 按键 A 得到 getevent 信息

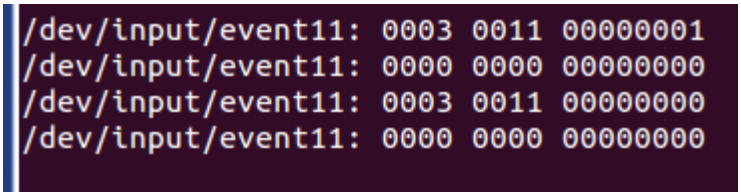
右摇杆的上下方向扳动，从 getevent 中得到 axis id 为 0x5，对应 kl 文件中的

axis 0x05 RZ，两个摇杆的其他方向及扳机键的模拟量部分(扳机键既有 BUTTON 事件又有 axis 事件)同理。



12-9 右摇杆上下方向扳动得到 getevent 信息

十字键的上下方向扳动，从 getevent 中 id 为 0x11，对应 axis 0x11 HAT_Y，十字键的左右方向同理。



12-10 十字键上下方向扳动得到 getevent 信息

4）部分按键有多重功能，比如 A 键除了 BUTTON_A 事件还有 DPAD_CENTER 事件，B 键除了 BUTTON_B 事件还有 DPAD_BACK 事件功能，需要确认是否需要提供对应的 Vendor_0101_Product_1d79.kcm 文件。

如果没有上述 kcm 的话会使用默认的 Generic.kcm，需要确认使用默认的 kcm 的情况下是否正常或者能否满足当前设备的需求，如果有需要参照 Generic.kcm 重新映射其键值即可。

```
479
480 ### Gamepad buttons ###
481
482 key BUTTON_A {
483     base:                               fallback DPAD_CENTER
484 }
485
486 key BUTTON_B {
487     base:                               fallback BACK
488 }
489
490 key BUTTON_C {
491     base:                               fallback DPAD_CENTER
492 }
493
494 key BUTTON_X {
495     base:                               fallback DEL
496 }
497
498 key BUTTON_Y {
499     base:                               fallback SPACE
500 }
501
502 key BUTTON_Z {
503     base:                               fallback DPAD_CENTER
504 }
505
506 key BUTTON_L1 {
507     base:                               none
508 }
509
510 key BUTTON_R1 {
511     base:                               none
512 }
513
514 key BUTTON_L2 {
515     base:                               none
516 }
517
518 key BUTTON_R2 {
519     base:                               none
520 }
521
522 key BUTTON_THUMBL {
523     base:                               fallback DPAD_CENTER
524 }
525
526 key BUTTON_THUMBR {
527     base:                               fallback DPAD_CENTER
528 }
529
530 key BUTTON_START {
531     base:                               fallback DPAD_CENTER
532 }
```

12-11 Generic.kcm 部分内容

5）新定义一个键值(keycode)，参考前面代码位置及基本功能描述第二部分列出来的相关文件即可。

12.3 摄像头

12.3.1 基本描述

目前常见的有四款摄像头：奥比摄像头、三合一新奥比摄像头、Eyesight 摄像头、游戏枪摄像头，还有两款不常见的摄像头：体感摄像头套件、水晶摄像头。

12-12 奥比摄像头

12-13 三合一新奥比摄像头

12-14 Eyesight 摄像头

12-15 游戏枪摄像头

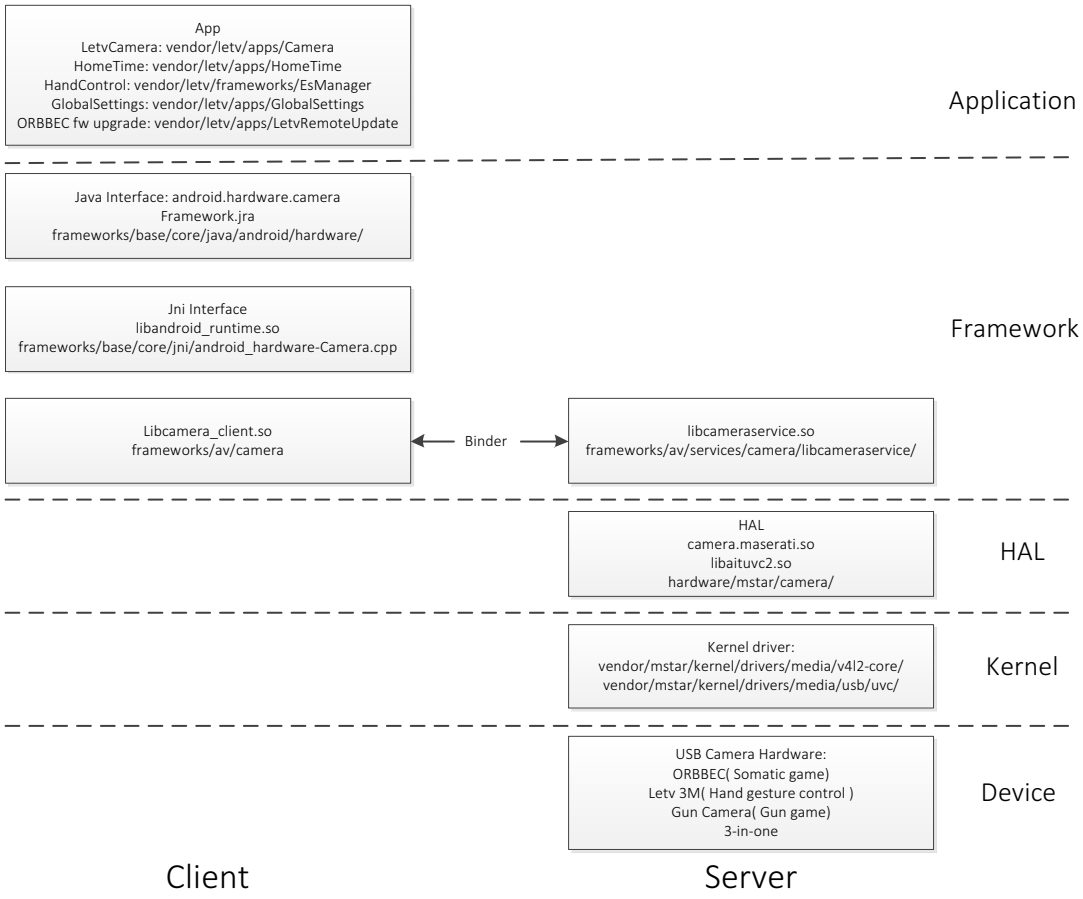
这几款摄像头的特性简述见下表：

摄像头	简述
奥比摄像头 (LeTMC-302)	3D 体感游戏，手势控制（握拳实现点击），拍照，录像，视频通话，（手势和乐拍及 HomeTime 不能同时使用，互斥的） 最大支持 720P, 801,918, 928, 8064,8094 跑 CTS 需要用这款摄像头
三合一奥比摄像头 (LeTMC-520)	3D 体感游戏，手势控制（握拳实现点击），拍照，录像，视频通话（手势和乐拍及 HomeTime 不能同时使用，互斥的） 最大支持 1080P, 938/8096 跑 CTS 需要用这款摄像头
Eyesight (Letv 3.5M)	手势控制（弯曲食指实现点击），拍照，录像，视频通话 最大支持 1080P，但是有显示问题， 还是限制到 720P
游戏枪摄像头 (TOLtech xeyez)	支持现代战争等枪感游戏，不支持手势 最大支持 720P
体感摄像头套件 （PRIMESENSE）	不支持拍照录像 只支持体感游戏和手势识别（点击动作是手掌前推，不是食指弯曲）
水星摄像头	拍照，录像，视频通话，不支持手势

12.3.2 工作原理

这里所有的摄像头都是符合 UVC(USB video class)硬件架构的设备，能够在不需要额外安装驱动的情况下即插即用，当然有个前提就是 linux kernel 中支持 V4L2(video for linux 2，支持视频采集及视频输出等功能的驱动架构，用来管理 UVC 设备的并且能够提供视频相关的一些 API)并打开了相关配置选项。

摄像头模块内部将光信号转换成数字数据，然后经 kernel、HAL、framework 到各种 app 进行不同的应用的示意图如下：



12-16 camera 架构示意图

12.3.3 代码位置及基本功能描述

1）vendor/mstar/kernel/drivers/media/v4l2-core/
vendor/mstar/kernel/drivers/media/usb/uvc/
摄像头 driver 的相关代码

2）hardware/mstar/camera/
Mstar 根据其平台硬件及编解码特性提供的摄像头 HAL 层的代码。

3）frameworks/av/services/camera/libcameraservice/
CameraService 的代码，作为 native 端向 binder 的 client 端提供 camera 控制的接口和能力。

4）frameworks/av/camera/
CameraService 的 proxy 端代码，向用户端提供接口。

5）frameworks/base/core/jni/android_hardware_Camera.cpp
Jni 部分的代码

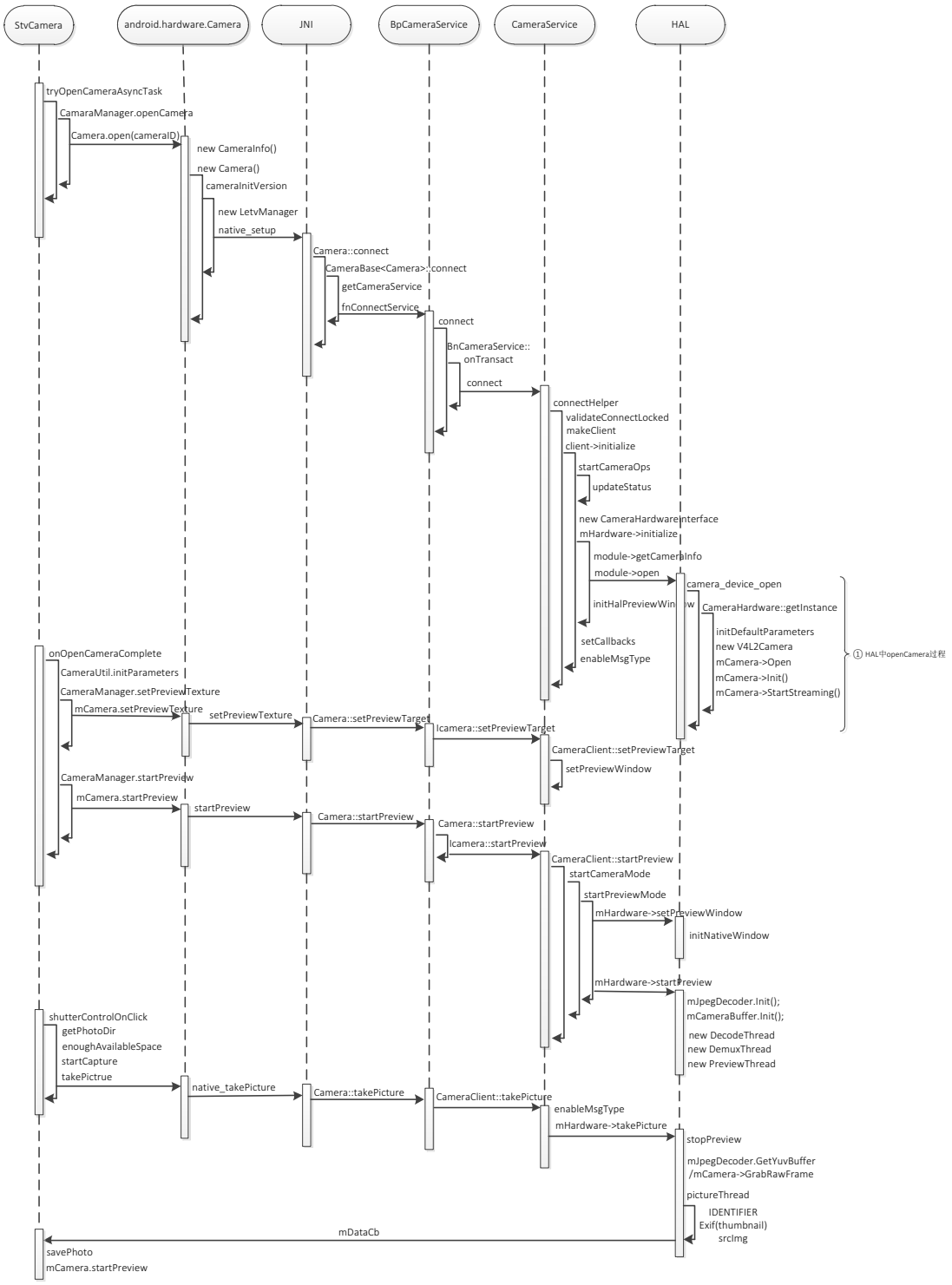
6）frameworks/base/core/java/android/hardware/
Framework 层 java 部分代码，向 app 提供接口

7）vendor/letv/apps/Camera

vendor/letv/apps/HomeTime/
vendor/letv/frameworks/EsManager/
vendor/letv/apps/GlobalSettings/
vendor/letv/apps/RemoterUpdate/
乐拍、HomeTime、Eyesight 摄像头体感服务、设置、fw 升级模块的相关代码

12.3.4 软件逻辑

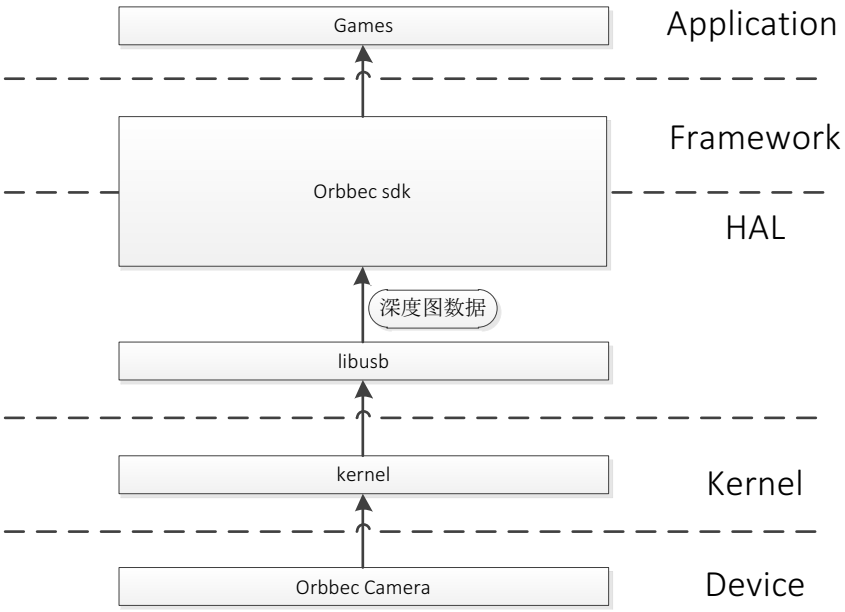
1)乐拍打开后至 preview 及拍照的大致代码流程作图如下：



12-17 preview 及 takePicture 流程图

2）奥比摄像头的体感功能只用到了深度图，奥比的 sdk 通过 libusb 接口读取奥比 camera 的深度图数据，然后在 sdk 中进行体感的识别并向游戏 app 提供接口。

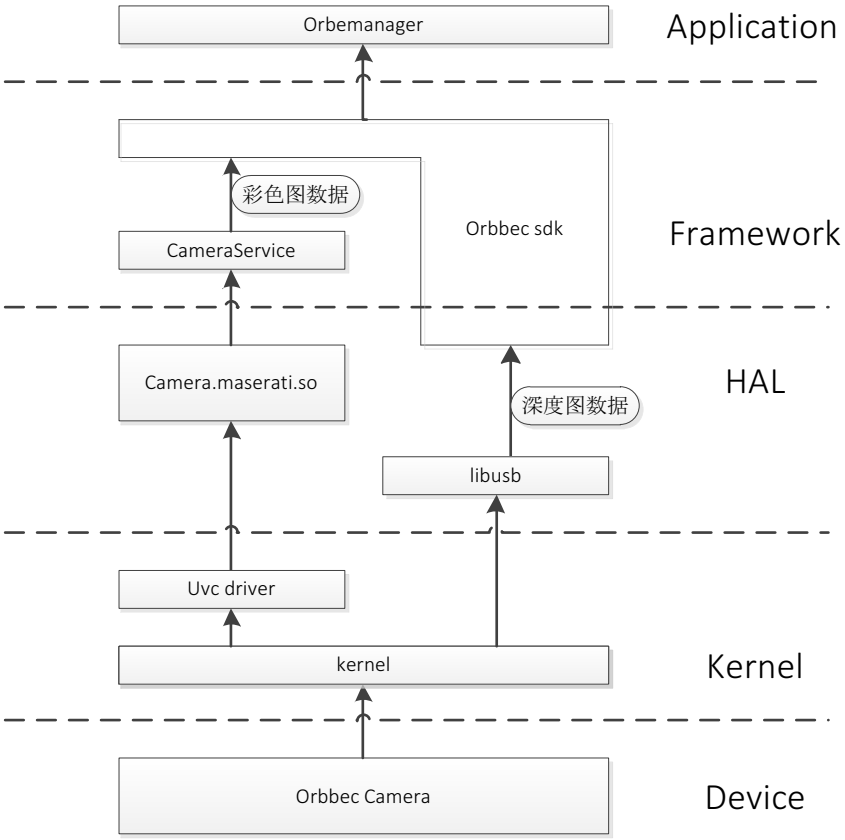
数据流走向如下图：



12-18 奥比摄像头体感数据流示意图

3）奥比摄像头的手势识别用到了深度图和彩色图，彩色图还是跟拍照类似的流程通过 camera API 拿到数据，深度图则是通过 sdk 读取，然后进行手势的识别并向系统注入事件。

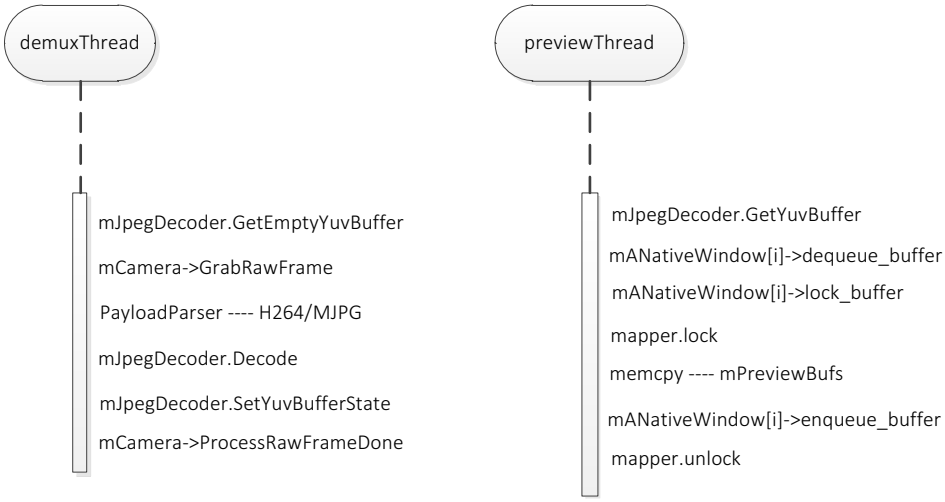
数据流走向如下图：



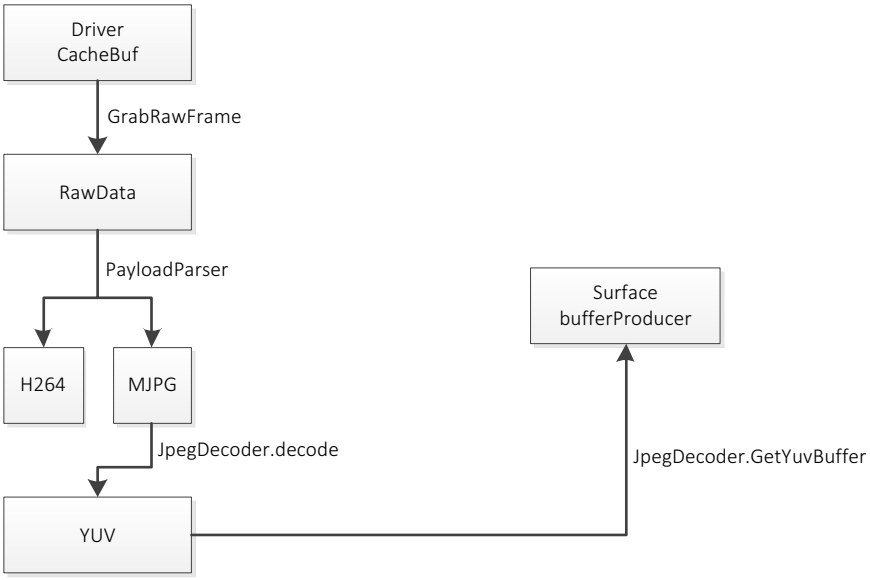
12-19 奥比摄像头手势识别数据流示意图

4）demuxThread 只在 Eyesight 摄像头使用时候才会用到，因为只有 Eyesight

摄像头获取到的视频数据是包含了不只一种格式的数据。以 Eyesight 摄像头为例，打开乐拍 preview 状态下 demuxThread 和 previewThread 的流程图及数据流如下：

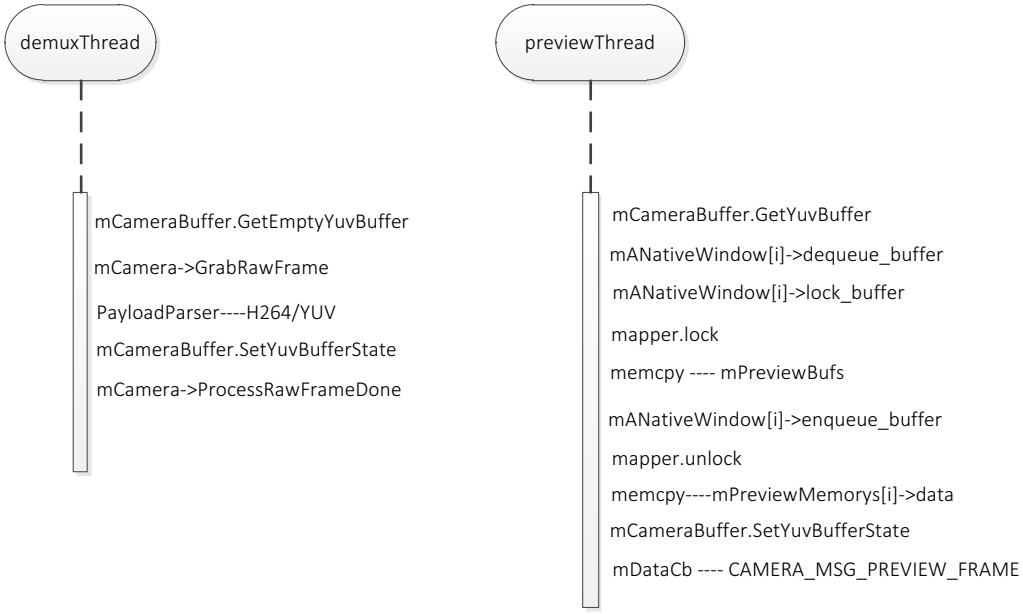


12-20 乐拍工作中 demuxThread 及 previewThread 流程图

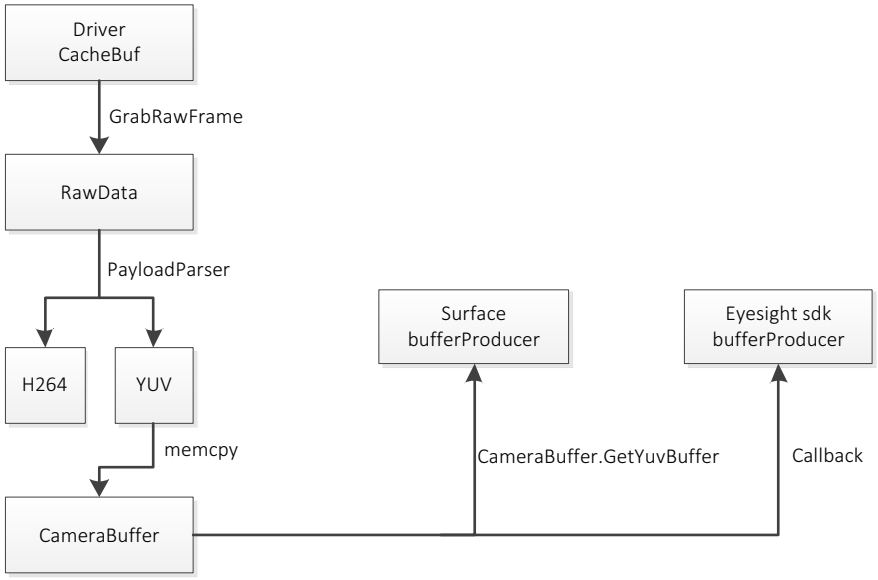


12-21 乐拍工作中数据流走向图

5）以 Eyesight 摄像头为例，开启手势识别状态下 demuxThread 和 previewThread 的流程图及数据流如下：

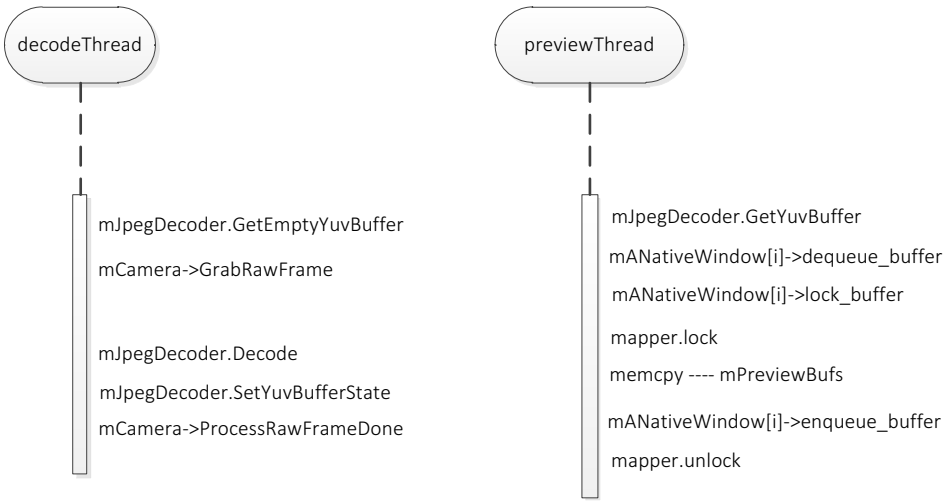


12-22 手势识别工作中 demuxThread 及 previewThread 流程图

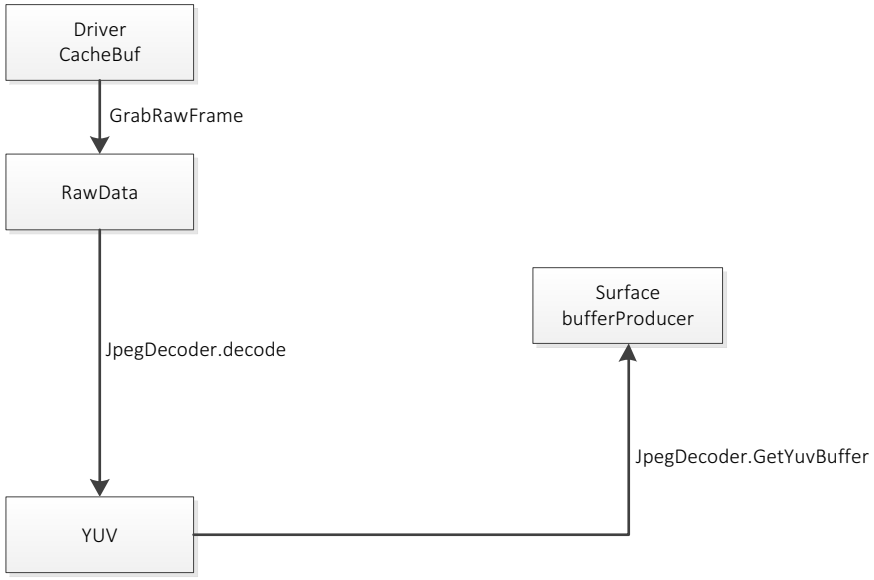


12-23 手势识别工作中数据流走向图

6）以奥比三合一摄像头为例，打开乐拍 preview 状态下 decodeThread 和 previewThread 的流程图及数据流如下：

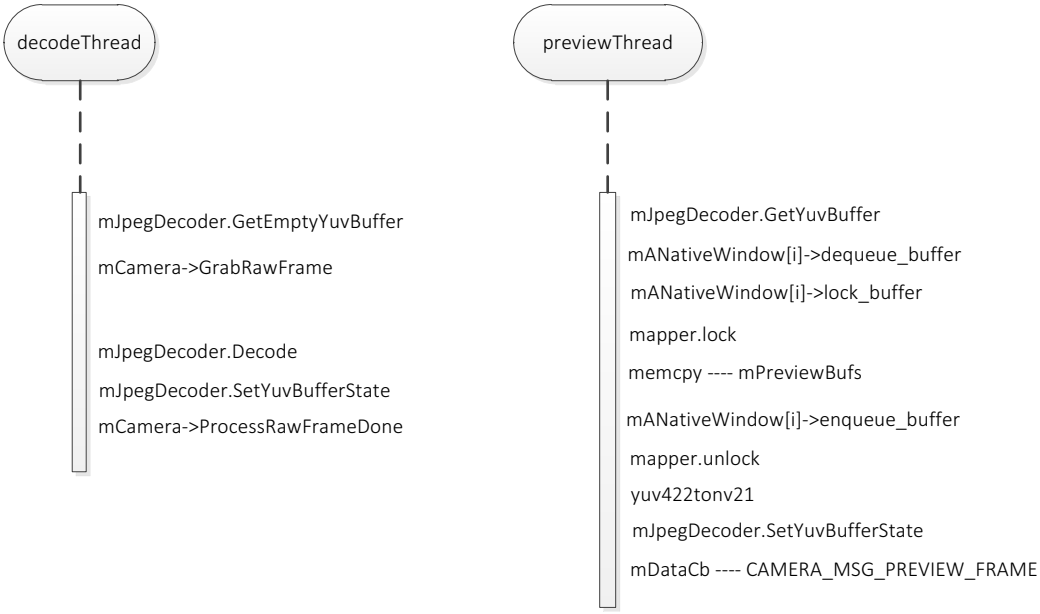


12-24 乐拍工作中 decodeThread 及 previewThread 流程图

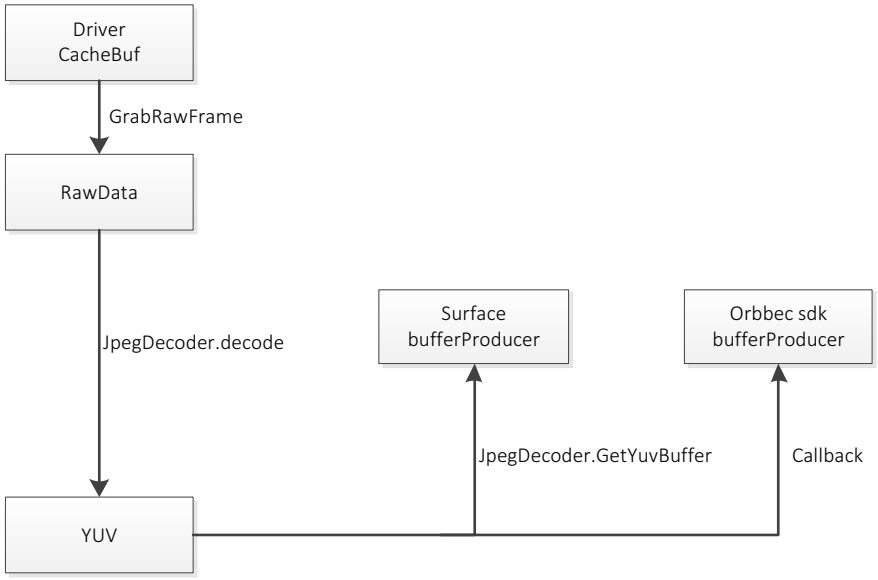


12-25 乐拍工作中数据流走向图

7）以奥比三合一摄像头为例，开启手势识别状态下 decodeThread 和 previewThread 的流程图及数据流如下：



12-26 手势识别工作中 decodeThread 及 previewThread 流程图



12-27 手势识别工作中数据流走向图